

PosPrintFlex ver. 1.0.0 Library
User Manual

Introduction.....	2
Connection options and used hardware	3
Matrix printer connected via COM interface.....	3
An ink-jet printer with built - in IrDA interface	4
A matrix printer connected by means of IrDA-COM adapter	4
A matrix printer connected via a cradle with built-in IrDA-COM adapter	5
Usage of solution POSPack	6
PosPrintFlex API	7
Functions.....	7
ppfVersion.....	7
ppfSetup	7
ppfStartDocument	7
ppfEndDocument	7
ppfStartPage.....	8
ppfEndPage	8
ppfPrintText	8
ppfPrintLine	9
Structures	9
ppfSettings	9
Examples of usage PosPrintFlex API	12

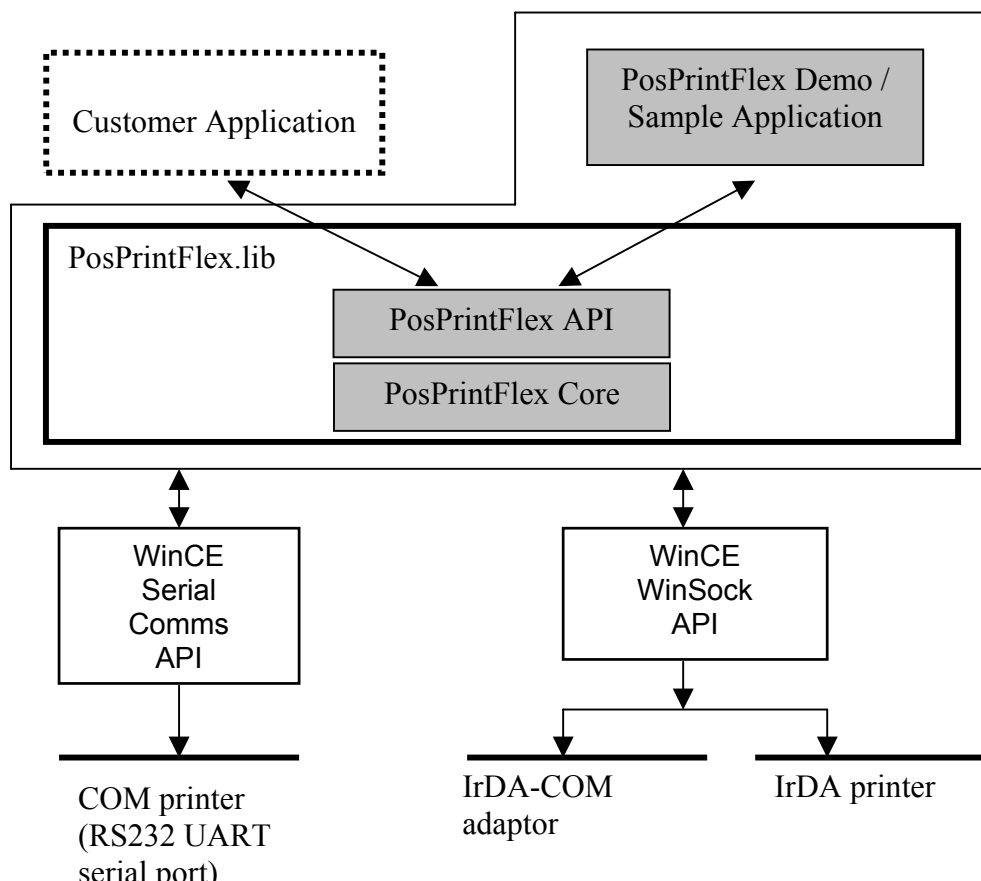
Introduction

PosPrintFlex is an easy-to-use tool of a software developer, allowing one to add printing capability (on matrix and POS printers) for hand-holds operating under Pocket PC /Pocket PC 2002 operating systems.

Key features of **PosPrintFlex** library, which favourably distinguish it from other libraries of the similar schedule and make it more lucrative, are:

- Simple-in-use library engine giving to the designer a print-out capability of text lines, as well as control of the text formatting and paginating.
- Simplicity of connection and usage of the library in your applications.
- Printing of any documents, such as invoices, receipts, load reports and others.
- Operation with matrix printers that allows one a fast and easy print-out of up to 5 full-page copies of the same document in "field" on-the-go conditions.

PosPrintFlex is a program library written in C language, which allows the user's application to make printing on IBM- and Epson-compatible printers connected to a Pocket PC via COM or infrared (IrDA) interfaces.



PosPrintFlex is the library unit and the initial code in C language, developed for Pocket PC of applications and supporting the Pocket PC and Pocket PC 2002 operating systems.

Connection options and used hardware

PosPrintFlex can be used in the various configurations and different hardware. For the sake of brevity, only few options of usage of the given library with various equipment are listed below.

Matrix printer connected via COM interface

Let's discuss the two main options for Pocket PC connection to a standard matrix printer with COM the interface:

1. A pocket computer with a COM port is connected directly to the printer with usage of a special cable. We shall illustrate the given connection type by the example of handheld computer Compaq iPAQ and printer Epson LX 300 +.



The required cable for the given connection can be ordered directly from our company.



2. By using a standard cable, printer is connected to COM port of Pocket PC cradle. We shall illustrate the given connection type by an example of handheld computer RoverPC P5 and printer OKI Microline 280.



For the given sort of connection in *PosPrintFlex* library the type of printer PRINTER_COM is used.

An ink-jet printer with built - in IrDA interface

The pocket computer, with help of IrDA interface is connected directly to the printer with built - in IrDA interface. We shall illustrate the given connection type by an example of handheld computer Dell Axim 5 and printer Canon BJC-85.



For the given type of connection in *PosPrintFlex* library the type of printer PRINTER_IRDA_CLIENT is normally used.

A matrix printer connected by means of IrDA-COM adapter

IrDA adapter with built-in support of conversion for RS232 port into IrDA (which does not require any software for usage of a special IrDA protocol) is connected to COM port of a standard matrix printer. The Pocket PC with help of IrDA interface is connected to IrDA-COM adapter. We shall illustrate the given connection type by the example of handheld computer Casio Cassiopeia E-125, IrDA-COM ACTiSYS ACT-IR100S adapter and matrix printer OKI Microline 280.



For the given sort of connection in *PosPrintFlex* library the type of printer `PRINTER_IRDA_CLIENT` is normally used.

A matrix printer connected via a cradle with built-in IrDA-COM adapter



PDASet Software

<http://www.pdaset.com>

The printer is connected by a standard cable to COM port on a Pocket PC cradle, which, in its turn, is connected to the respective Pocket PC through IrDA interface. We shall illustrate the given connection type by an example of handheld computer Casio Cassiopeia IT-70M30E and printer Epson LX 300 +.

For the given type of connection in ***PosPrintFlex*** library the type of printer PRINTER_IRDA_SERVER (as in the example considered above), and PRINTER_IRDA_CLIENT for other type of equipment can be used.

Usage of solution POSPack

PosPrintFlex can also be applied within framework a ready-made solution POSPack represented on the site <http://www.pospack.com/>.



For the given sort of connection in ***PosPrintFlex*** library the type of printer PRINTER_COM is used.

PosPrintFlex API

Functions

ppfVersion

ppfVersion function returns number of the current version of **PosPrintFlex** library.

DWORD ppfVersion (VOID);

Parameters

This function has no parameters.

Return value

If the function is executed successfully, the return value such as **DWORD**, contains high and low numbers of versions of the library.

ppfSetup

ppfSetup function installs current customizations of the printer.

VOID ppfSetup (ppfSettings setup);

Parameters

setup

The structure containing settings of the printer.

Return value

The function has no return values.

ppfStartDocument

ppfStartDocument function installs connection to the printer, initializes the printer and transferred it into a ready-to-print mode.

BOOL ppfStartDocument (VOID);

Parameters

This function has no parameters.

Return value

If the function is executed successfully, return value is not equal to zero.

If the function is executed unsuccessfully, return value is equal to zero.

Remarks

The function uses a Control sequence **printerInitSequence**, current installations of the printer.

ppfEndDocument

ppfEndDocument function completes printing the document and closes connection to the printer.

BOOL ppfEndDocument (VOID);

Parameters

This function has no parameters.

Return value

If the function is executed successfully, return value is not equal to zero.

If the function failed to be executed successfully, return value is equal to zero.

Remarks

The function uses a Control sequence **printerCleanupSequence**, current installations of the printer.

ppfStartPage

ppfStartPage function transferred the printer into a ready-to-print mode for printing new page.

BOOL ppfStartPage (VOID);

Parameters

This function has no parameters.

Return value

If the function is executed successfully, return value is not equal to zero.

If the function is executed unsuccessfully, return value is equal to zero.

Remarks

The function uses a Control sequence **newPageSequence**, current installations of the printer.

ppfEndPage

ppfEndPage function completes printing of the current page. For printing a new page, it is necessary to call **ppfStartPage** function.

BOOL ppfEndPage (VOID);

Parameters

This function has no parameters.

Return value

If the function is executed successfully, return value is not equal to zero.

If the function is executed unsuccessfully, return value is equal to zero.

Remarks

The function uses a Control sequence **endPageSequence**, current installations of the printer.

ppfPrintText

ppfPrintText function prints out the text, using preset formatting.

```
BOOL ppfPrintText (char *beginSequence,  
                  unsigned beginSequenceLength,  
                  char *string,  
                  char *endSequence,  
                  unsigned endSequenceLength);
```

Parameters

beginSequence

The initial Control sequence, which provides initialization of the selected text formatting.

beginSequenceLength

Length of an initial Control sequence in bytes.

string

Address of the string ended with a null character, which is printed out.

endSequence

The completing Control sequence initializing a new or resetting the selected text formatting.

endSequenceLength

Length of a completing Control sequence in bytes.

Return value

If the function is executed successfully, return value is not equal to zero.

If the function is executed unsuccessfully, return value is equal to zero.

ppfPrintLine

ppfPrintLine function prints out the text, using current formatting, and transfers to a new string upon termination of the text print-out.

```
BOOL ppfPrintLine (char *string);
```

Parameters

string

The pointer on a string ending with a null character, which is printed out.

Return value

If the function is executed successfully, return value is not equal to zero.

If the function is executed unsuccessfully, return value is equal to zero.

Remarks

The function uses a Control sequence **newLineSequence**, current installations of the printer.

Structures

ppfSettings

Contains the information used for customization, initialization and handle of the printer. It is used by **ppfSetup** function.

```
typedef struct ppfSettings {
```

```
    PrinterType    printerType;  
    char           portName [MAX_PORT_NAME_LENGTH + 1];
```

PRINTABORTPROC abortProc;

char printerInitSequence [MAX_SEQUENCE_LENGTH];
unsigned printerInitSequenceLength;
char printerCleanupSequence [MAX_SEQUENCE_LENGTH];
unsigned printerCleanupSequenceLength;

char newPageSequence [MAX_SEQUENCE_LENGTH];
unsigned newPageSequenceLength;
char endPageSequence [MAX_SEQUENCE_LENGTH];
unsigned endPageSequenceLength;

char newLineSequence [MAX_SEQUENCE_LENGTH];
unsigned newLineSequenceLength;

} ppfSettings;

Members

printerType

The used printer type.

At present, only the following three types of printers are supported:

PRINTER_COM	Printer is connected to a handheld (Pocket PC) via COM interface.
PRINTER_IRDA_CLIENT	Printer is connected to a handheld (Pocket PC) via IrDA interface. Connection with remote IrDA device is initiated by the handheld.
PRINTER_IRDA_SERVER	Printer is connected to a handheld computer (Pocket PC) via IrDA interface. Connection with the remote IrDA is initiated by the remote device.

portName

Name of a port, to which the printer is connected, for example "COM1:". For the printers connected through infrared (IrDA) interface, a name of a port corresponds to "IrDA:IrCOMM".

abortProc

The pointer to the function, which resets the print-out termination flag. The process of printing proceeds until the Return value **abortProc** function attains a zero value or until print-out would be terminated by **ppfEndDocument** function.

If value of the pointer on **abortProc** function equals to **NULL**, then a control of printing is provided by the library built-in engine.

printerInitSequence

Control sequence of initialization of the printer. The given sequence is used by **ppfStartDocument** function.

printerInitSequenceLength

Length of the control sequences of initialization of the printer in bytes.

printerCleanupSequence

Control sequence for termination of work with the printer. The given sequence is used by **ppfEndDocument** function.

printerCleanupSequenceLength

Length of control sequences for termination of work with the printer in bytes.

newPageSequence

Control sequence of initialization of a new page. The given sequence is used by **ppfStartPage** function.

newPageSequenceLength

Length of the control sequences of initialization of a new page in bytes.

endPageSequence

Control sequence of the current page print-out completion. The given sequence is used by **ppfEndPage** function.

endPageSequenceLength

Length of the control sequence of the current page print-out completion in bytes.

newLineSequence

Control sequence of word wrap-around. The given sequence is used by **ppfPrintLine** function.

newLineSequenceLength

Length of the control sequences of word wrap-around in bytes.

Examples of usage PosPrintFlex API

Examples of connection and usage ***PosPrintFlex*** library in the application are in folder **DEMO** and represent the following:

1. ***Demo1*** application is an example of a simple application of the library for printing a line “Hello world!” with various formatting supported by IBM- and Epson-compatible printers.
2. ***Demo2*** application is an example of printing of the multipart document with complex formatting, e.g. an invoice.

Both examples are developed with usage of the integrated developer’s environment Microsoft eMbedded Visual C ++ 3.0.